Modelling Project Coordination in a Multi-Agent Framework

Brazier, F.M.; Jonker, C.M.; Treur, J.

published in
Proc. of the Fifth Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, WET ICE'96
1996

document version
Publisher's PDF, also known as Version of record

Modelling Project Coordination in a Multi-Agent Framework*

F.M.T. Brazier, C.M. Jonker, J. Treur

Vrije Universiteit Amsterdam
Department of Mathematics and Computer Science
Artificial Intelligence Group
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
Fax: +31.20.4447653 Email: {frances,jonker,treur}@cs.vu.nl

1 Introduction

Distributed project coordination requires insight in the types of interaction involved in engineering practice. In current practice, well-structured hierarchical management and decentralised project organisation are often combined. Within an organisation, a number of levels can be found within which responsibility for effective interaction is delegated to the engineers themselves. Engineers decide when to exchange preliminary ideas and partial designs, when to acknowledge possible conflicts and when to resolve such conflicts, when to question requirements, et cetera. A combination of traditional management structures and virtual organisations result in dynamic structures, liable to considerable change during the life span of a project.

The types of interaction encountered in such real-life engineering situations show how intricate such processes can be. Within the multi-agent community the problem of distributed problem solving has been recognised; see for example (Dunskus, Grecu, Brown and Berker, 1995; Petrie, 1994). In (Jennings, 1995) an informal multi-agent model for cooperative problem solving is proposed. Essential elements of this model are the dynamic organisation and management of joint activities, susceptive to change due to unexpected events. As described, the model, however, does not provide enough detail to support analysis, modelling and implementation of design coordination systems in specific domains. To acquire a more precise description of this model more detailed analysis is required.

In this paper a real-life design project is analysed for a situation in which traditional management and virtual organisations are combined: the design of part of the interior of a specific aircraft. The DESIRE framework (Langevelde, Philipsen and Treur, 1992; Brazier, Dunin-Keplicz, Jennings and Treur, 1995; Brazier, Treur, Wijngaards and Willems, 1995, 1996) is used to model the coordination of this cooperative distributed design project, using Jennings' model as a frame of reference.

2 Distributed project coordination

Coordination of complex engineering projects often entails coordination of individuals but also coordination of groups, often somehow related to departments and/or project groups. These entities, whether departments, project groups or individuals, may be modelled as agents: each with their own responsibilities and autonomy.

2.1 Coordination of engineering projects

Project coordination occurs at many levels: often groups of managers interact on a regular basis, to evaluate the current situation with respect to availability of resources (e.g., technology, manpower, material, expertise), planning, integration, et cetera. This holds for all phases of engineering: during the initial design of a concept, feasibility studies, design definition, full scale development, validation, et cetera. In essence, design entails consideration and refinement of requirements, process coordination and modification and refinement of a design object description (see Brazier, Langen, Ruttkay, Treur (1994) for a generic model of design), during each of these phases across disciplines. When and how managers really interact, however, most often depends on the design problems encountered, and on their willingness to cooperate.

In this paper a simplified example of the coordination of a routine design project is used for the purpose of illustration: the design of aircraft interior. Agents refer to individuals (or groups of individuals) with a specific task in the project. Requirements are specified at the level of detail required for verification, including specification of the verification procedures.

A design project manager is assigned the task of coordinating all design activities for the interior of an aircraft, for example the design of the toilet unit, luggage bins, wardrobe, gallies, side panels, and the floors, often in close collaboration with the financial department. The responsibility for the design of each of the individual units is delegated to a unit manager, who in turn coordinates the design of more specific aspects of that unit. The design project manager interacts with a number of specialists: financial specialists, styling specialists, logistic specialists, tooling specialists, et cetera, to coordinate the project as a whole. At this level, coordination is clearly hierarchically organised. Although relatively well-defined, the frequency and content of interaction and cooperation is not as easily specified.

Detailed design at the level of one of the units, however, will be used to illustrate the value of our approach. The unit manager considered receives requirements for the aircraft as a whole, together with technical specifications for a specific unit, in our example the toilet unit. He/she is responsible for the integrated design of the unit, but also for interaction with other unit managers and the project manager, in particular with respect to control and configuration management. The unit manager coordinates detailed design of the unit: he/she examines (partial) designs produced by design engineers, electrical engineers, and systems engineers, identifies inconsistencies, and interacts with the designers/engineers to find solutions. The unit manager is responsible for the provision of information within his/her unit group: the most recent version of the integrated design, relevant guidelines and decisions taken within the project management group, et cetera.

The engineers, in turn, coordinate their own design processes. Design engineers, for example, interact not only with electrical engineers and systems engineers, but also with other experts, such as product specialists, purchasing department, tooling specialists and styling specialists, when necessary. When and how other specialists are involved, is left up to the discretion of the individual engineers: they themselves 'defend' virtual organisations

For the sake of simplicity, the above example of design will be modelled for one unit, with one engineer of each signature. These engineers will most often represent a group of engineers responsible for the tasks assigned in this model. The patterns of communication between engineers are, however, comparable.

2.2 Design process

Four main activities can be distinguished within this design process: verification, product design, product selection and product definition. The sequence of product design, selection, and definition is, in principle, an iterative process. Verification requirements imposed on the unit (both generic and specific) are analysed and translated into verification means, methods and tools for specific elements of the design product and/or production process during product design.

Product design entails design of (1) the design product, and (2) the production process: alternative products and processes are proposed and analysed. The sequence within which parts of the product and parts of the production process are considered, depends largely on expectations with respect to time needed to acquire materials, and to design and manufacture tooling. One of the main requirements on the design process is to make information

* Brazier, F.M.T., C.M. Jonker & J. Treur (1996). Modelling Project Coordination in a Multi-Agent Framework. In: Proceedings of the Fifth Workshops on Enabling Technology for Colloborative Enterprises, WET ICE'96, IEEE Computer Society Press, 1996, pp. 148-155.

on both of these aspects available (tooling and supplies) as soon as possible.

Product selection involves interaction with one or more designers/engineers, and specialists depending on the level of detail involved, both for the design product and the production process. Factors such as intricacy of manufacturing process, cost, maintainability, standardisation, weight, are of importance in this phase. Product definition defines the design of the product and the production process at the level of detail required for manufacturing. Effect and efficiency are examined, in particular for recurring processes within the production process, and for recurring parts.

3 Specification of Multi-Agent Systems

In projects such as the design project sketched above task coordination between agents is essential. As agents, however, often perform more than one task, (sequentially or in parallel), task coordination within the agents themselves is also of importance. Within the formal compositional framework DESIRE (Langevelde, Philipsen and Treur, 1992; Brazier, Treur, Wijngaards and Willems, 1995; Brazier, Dunin-Keplicz, Jennings and Treur, 1995, 1996) task models are used to define compositional architectures. Task models include knowledge of

(1) a task (de)composition,
(2) information exchange,
(3) sequencing of (sub)tasks,
(4) sub-task delegation, and
(5) knowledge structures,

These five types of knowledge are explicitly modelled and specified at different levels of abstraction. Tasks are defined at different levels of abstraction, resulting in a task (de)composition. Different levels of abstraction are distinguished within knowledge structures; for example taxonomies of information types. Tasks refer to these knowledge structures. Sequencing of tasks and goals, and information exchange reflect the abstraction level of tasks involved. Task delegation, the last of the five types of knowledge, is also defined at all levels of abstraction within a task model. More abstract tasks may be delegated to more than one party, whereas more specific tasks are often delegated to one particular party.

The model of cooperation presented in this paper has been formally specified within the DESIRE framework. The semantics of the formal specification language are well-defined, based on temporal logic; see (Brazier, Treur, Wijngaards and Willems, 1996). By explicitly modelling and specifying the semantics of static and dynamic aspects of a system, a well-defined conceptual description is acquired that can be used for verification and validation, but also is a basis for reuse. Translation to an operational system is straightforward; the framework, in fact, includes implementation generators with which formal specifications can be translated into executable code. DESIRE has been successfully applied to design and develop both single agent and multi-agent systems (Brazier and Treur, 1994; Brazier, Dunin-Keplicz, Jennings and Treur, 1995; Dunin-Keplicz and Treur, 1995).

4 A Model of Cooperation

To successfully develop a support system for cooperation in a complex, dynamic and not always predictable environment, a well-defined and transparent model of cooperation is required: a model that is robust and flexible enough to cope with unexpected events. To this aim in (Jennings, 1995) a model for cooperative problem solving using joint intentions was introduced, based on experience in industrial applications. The model describes both the phase of setting up (organising) a joint project and the phase of performing the joint project, including the management of unexpected difficulties. In (Jennings, 1995) details of this model are described for an implementation in one specific environment. This limits possibilities for reuse of the model. In this section, the cooperation model is described in terms of specifications at the conceptual level in the compositional framework DESIRE. For detailed specifications see (Brazier, Jonker and Treur, 1996).

In Jennings' model of cooperation agents are capable of organising projects. An agent decides to organise a project to reach a given goal (in the example of Section 2, the goal of the design project organised by the design manager (DPM) is to design the interior of the aircraft, in particular the design of the toilet unit). With respect to the current state of the world, an agent determines a set of activities to reach this goal and the temporal dependencies between the activities. The DPM, for instance, considers dependencies between activities such as coordination, design of the construction, design of electrical systems, design of other systems, styling, and tooling. The organising agent then identifies other agents capable of performing the activities (a unit manager UM, a design engineer DE, an electrical engineer EE, a systems engineer SE, a styling specialist SS, a tooling expert T, et cetera). In interaction with these agents, the organising agent determines which agents are willing and able to participate in the project. On the basis of this information, the activities to be performed, the order in which the activities are to be performed and the deadline, the organising agent tries to put together a project team and a project schedule (called a *recipe*). The creation of this recipe is an iterative process requiring interaction with the other agents on their own schedules (related to other projects). When completed, the recipe is sent to all participants, and the project commences.

Once committed, each participating agent (including the organiser) receives the final recipe, and is committed to the relevant time interval in the recipe. Each agent has the same obligation towards the project: each member monitors the progress of the project and is equally responsible for its success. If a team-member discovers a problem that endangers the project, he/she informs all participants. One of the agents (e.g., the project manager) can then take the initiative to modify the project plan, to create a new project for the same goal or to inform all participants that the goal is unattainable or that it is no longer necessary to reach the goal.

In Figure 1 a hierarchical task decomposition for a cooperative agent is depicted. In Sections 4.1 to 4.8 the components that correspond to tasks in the hierarchy are described in more detail.

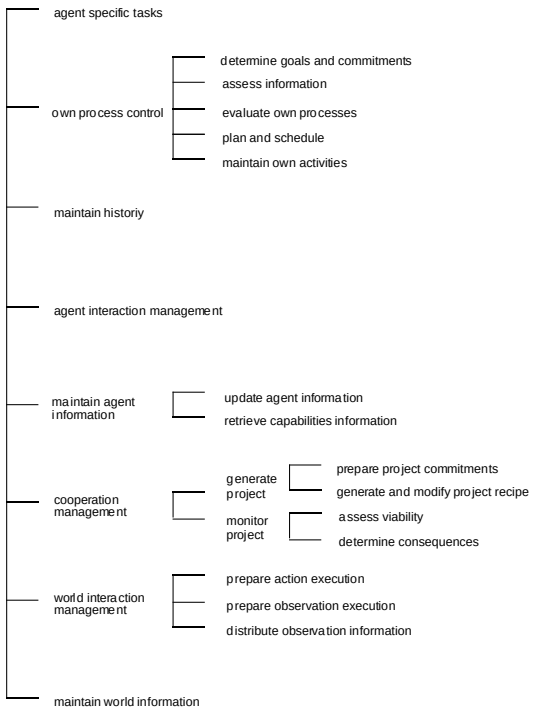

Figure 1 Task hierarchy for a generic agent

A cooperative agent performs a number of generic tasks. Some of these tasks deal with the relationship of an agent to the world: maintaining information about the world (*world model*), and managing interaction with the world (*observation, execution of actions* that change the world). Other tasks concern its relationship to other agents: maintaining information on other agents (*agent models*), managing interaction with other agents (*communication*), and managing activities performed jointly with other agents (*cooperation*). Furthermore, tasks of a more reflective nature are

performed: maintaining information of an agent's own processes over time (*history*), and managing an agent's own processes (*own process control*). In addition to these generic tasks, agent specific tasks are distinguished: tasks that may differ between agents (*agent specific tasks*). A graphical representation of these tasks is shown in Figure 1.

Each of the tasks depicted in Figure 1 can be described in more detail. The eight components responsible for the tasks agent_specific_tasks, own_process_control, maintain_history, agent_interaction_management, maintain_agent_information, cooperation_management, world_interaction_management, and maintain_world_information are presented below.

4.1 Agent Specific Tasks (AST)

AST is a composed component that is mostly domain-specific and that may differ per agent. It contains a task-hierarchy and knowledge necessary to perform tasks in interaction with other components of the same agent.

4.2 Own Process Control (OPC)

The agent component OPC is a composed component responsible for determining, planning, scheduling and monitoring an agent's activities. Furthermore, it is responsible for maintaining all relevant information on the agent's activities and its status. These sub-tasks are performed by OPC's sub-components: determine_goals_and_commitments (DPC), assess_information (AI), evaluate_own_processes (EOP), plan_and_schedule (PS) and maintain_own_activities (MOA).

4.2.1 Determine Goals and Commitments (DGC)

DGC determines goals of an agent on the basis of its motivations, priorities, and deadlines and its role within a system. Selection of a goal depends on motivation: motivation is a necessary precondition for goal selection. Selection of a goal implies individual commitment to the goal.

4.2.2 Assess Information (AI)

The AI component maintains all relevant information on an agent's activities: which information is based on its own observations; which on own assumptions; which has been received by communication, and from which source; and which information has been derived, and is based on which other information.

4.2.3 Evaluate Own Processes (EOP)

This component is responsible for the evaluation of the progress of an agent's activities with respect to its individual commitments. It involves monitoring relevant activities (its own and other agents) and analysing monitoring information. During analysis EOP may, for example, deduce that the motivation for a goal has disappeared: this goal is then removed.

4.2.4 Plan and Schedule (PS)

The component PS is responsible for planning and scheduling an agent's activities, upon request for participation in a project by another agent or on the basis of information received from EOP or DGC. The component PS uses domain-knowledge to find a set A of activities, called a plan, that meets the following criteria: (1) execution of the plan will lead to the fullfilment of a goal G, (2) the plan can be scheduled without contradicting prior commitments, (3) the plan matches the priority and the deadline of the goal. If no such plan and schedule can be found, not even by requesting the help of other agents, this must be communicated to EOP. Another goal can then be selected by DGC. If an agent cannot reach the goal G itself while respecting the priority and deadline, but the goal may possibly be reached with the help of others, then all relevant information is sent to CM, which will try to create a project to reach the goal.

4.2.5 Maintain Own Activities (MOA)

This component stores an agent's own schedule, which actions an agent can perform (domain dependent) and which commitments an agent has made to which goals. Commitments can be made with respect to other agents and projects.

4.3 Maintain History (MH)

The component MH is responsible for the storage of the sequences of internal and external processes of an agent, for which purposes and with which results. Upon request, part of this information can be sent to other components or agents. Information of this kind is useful in strategic reasoning. For example, if a goal can be reached via different recipes and one of these recipes has previously been attempted and failed, another recipe should be attempted.

4.4 Agent Interaction Management (AIM)

The component AIM manages communication with other agents, in particular with team members of a project. It receives information from CM which it transfers to (possible) participants in a project. Furthermore, it receives (communicated) information from other agents which it transfers to other relevant components. For example, upon receiving a new recipe, AIM determines the subset of recipe-elements that concern its own activities. This subset is passed on as "own process" information to OPC. The whole recipe is sent to CM.

4.5 Maintain Agent Information (MAI)

Upon request MAI provides other agents or other sub-components with names of agents capable of performing certain specified activities. Two sub-components are responsible for the performance of this task: update_agent_information (UAI) and retrieve_capabilities_information (RCI).

4.5.1 Update Agent Information (UAI)

UAI maintains models of other agents known to an agent itself. A model of another agent consists of statements that express cooperativeness of the other agent, its availability (that it normally has no time to help other agents, or normally is able to help), punctuality with respect to deadlines, et cetera. UAI stores and updates its knowledge by maintaining which activities other agents are capable of performing, the projects in which they participate and the goals to which they are committed.

4.5.2 Retrieve Capabilities Information (RCI)

RCI provides, for each activity, the names of all agents known to be capable of performing an activity and the available meta-information concerning the exhaustiveness of the information.

4.6 Cooperation Management (CM)

The component CM is a composed component responsible for all tasks concerning projects, project commitments and cooperation.

4.6.1 Generate Project (GP)

Given the goal G, motivation M, priority p, deadline T, all possible sets A of activities with which goal G can be reached, and an agent's own capabilities, the component GP has two main tasks: to prepare project commitments, and to generate and modify project recipes.

The component **Prepare Project Commitments (PPC)** determines a preferred set A of activities with which goal G can be reached. Using domain-knowledge the dependencies between the activities in A are determined using critical path methods. This (partial) ordering of the activities in A is important in the development of a recipe R for goal G. Given this dependency-graph PPC determines which agents can and are willing to perform activities to help reach goal G. The dependency-graph for A, the information (G, M, p, T), the relevant capabilities of the willing participants (including the agent's own relevant capabilities) and the corresponding names of the agents, are sent to GMR.

Using PPC's information, the component **Generate and Modify project Recipe (GMR)** designs a recipe R that conforms to the interdependencies between the activities in A (thus leading to G's fulfilment). The recipe R is interactively designed by iteratively generating and sending proposed recipe elements to agents interested in participation. A recipe element consists of a task of A, a willing participant capable of performing that task, a priority p and a deadline T for that task. The willing participants accept, adapt or reject the proposed recipe elements. Acceptance or adaptation of a recipe element implies that the agent commits itself to this element. GMR adjusts the partial recipe depending on the replies from participating agents. A recipe may be found that is acceptable to all participants and that will reach goal G before its deadline. The duration of the recipe and team building is estimated on the basis of the number of activities involved, the number of willing participants and the time needed for communicating requests and responses. The time required for communication (depending on the situation) is assumed to be known. In addition,

communication is assumed to be error free. The resulting recipe is communicated to all participants.

4.6.2 Monitor Project (MP)

The component MP is responsible for the detection of the need for alterations to the project or the need to stop the project. MP monitors the progress of the project. In order to perform its task MP has two sub-components: assess_viability and determine_consequences.

Assess Viability (AV) monitors the viability and validity of the recipe. To check the validity of the project recipe, AV uses the same considerations as the sub-component evaluate_own_processes of the component own_process_control. To monitor the process it uses information received from OPC, WIM and MWI (its other components). It can also actively formulate requests for observational information from WIM, MWI or information of other agents via MAI and AIM.

Determine Consequences (DC) interprets AV's monitoring results. The component DC issues requests to find new recipes or to adapt existing recipes, to the component project_generation of CM and issues corresponding messages to the participants. DC also determines when a goal G should be withdrawn (for example, because the goal is unattainable, the goal has been reached, or because the motivation for the goal no longer exists) and prepares and issues a message to that effect to each participant.

4.7 Maintain World Information (MWI)

MWI contains the current world state as known to the agent. MWI stores all information obtained by monitoring the world (also the material aspects of all agents including the agent itself).

4.8 World Interaction Management (WIM)

The component WIM is responsible for the execution of observations and actions. An important sub-task of this component is the observation of the effects on the world of the tasks executed by the other agents and by the agent itself.

4.8.1 Prepare Action Execution (PAE)

This component prepares the execution of actions determined by AST by communicating to the world which actions should be taken.

4.8.2 Prepare Observation Execution (POE)

WIM prepares specific observations. The observational information is sent via DOI to those sub-components that analyse this information.

4.8.3 Distribute Observation Information (DOI)

Upon request, observational information is sent from DOI to other components (including MWI). DOI can also take the initiative to inform other components (including MWI) of (domain-dependent) important changes in the world.

5 Communication between agents

Interaction between agents is modelled by information links, controlled by the agent from which the links originate. Different types of information are exchanged through links: both *object level information*, such as information on the design object description, the initial cable routing, switch dimensions and positions, the initial design, product information, and *meta-level information*; i.e., requests for information, evaluation information on the design object description, conflicts between routing of cables and the initial design, and information on the design process (e.g., planning and scheduling). The double-arrowed lines in Figures 2 and 3 depict the information links that specify the exchange of these types of information between agents.

To describe the interaction between agents two scenarios will be sketched. First the creation of a project is sketched from the perspective of a design project manager in Section 5.1. A system trace is presented for the creation process, sketching the activation of agents, components of agents and the information communicated through time. As an example of project execution, the design of a unit is described from the perspective of a design engineer in Section 5.2, part of which is presented in a system trace.

5.1 Communication during project creation

In this section a scenario for project creation is described. The communication patterns are depicted in Figure 2. A trace of a process of project formation is shown in Table 4.

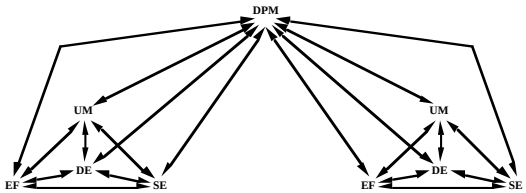

Figure 2 Communication during project creation

Project creation scenario

The component OPC of the design project manager (DPM) has the goal to design an aircraft (1). To reach this goal, DPM needs help. Thus, his component GP (part of CM) is activated to generate the project. Immediately, PPC (part of GP) is activated to determine which activities are needed to reach the goal and which possible team members for the project (2) can be found. For this purpose DPM requests possible participation from design engineers, electrical engineers, systems engineers, unit managers, styling specialists and tool experts. The requests are handled by DPM's AIM component (3). Each of these agents receives the request through its own AIM component (4), and considers the request for possible participation in its own component OPC (5). Each agent's AIM component returns an answer to the request (6). DPM receives the agents' responses (in his AIM component) (7). The replies are forwarded to the PPC component, which continues the preparation of project commitments in interaction with the possible participants (iterating steps 3 through 8). The information on the project activities and the willing participants is sent to GMR (part of GP). This component is responsible for the creation of the final recipe. This task involves frequent contact with the willing participants. Again this contact is handled by the AIM components of the agents (10,11). The OPCs of the willing participants check to see if the activities assigned to them fit in their own schedules (12). Information on the success or failure of their scheduling is sent by their AIM component (13) to the AIM component of DPM (14), which forwards it to GMR (15). By iterating steps 10 through 15, GMR creates a final recipe. The resulting recipe includes the global goal (i.e., aircraft to be designed given global requirements and specifications) and recipe elements. A recipe element related to the design of a unit includes the following information:

- the specific requirements and specifications for the unit to be designed (based on the initial design of the whole aircraft),
- one unit manager (UM),
- one design engineer (DE),
- one electrical engineer (EE), and
- one systems engineer (SE).

The resulting recipe is sent to each of the unit managers by AIM (16). The CM component of DPM makes sure that the resulting recipe will be monitored by its subcomponent MP (16).

After the unit groups have been formed the unit managers schedule the design process of their unit, following a similar pattern. For example, the unit manager responsible for the design of the toilet unit (i.e., toilet basin, counter top, sink and cabinet combination, et cetera) decides that the design engineer involved should make an initial design for the electrical engineer and the systems engineer. To ensure that the electrical engineer and the systems engineer can start as quickly as possible, the unit manager initially gives the toilet basin the highest priority compared to the top counter, the sink and the cabinet combination.

time point	agent	agent component	sub component	subsub-component
1.	DPM	OPC		
2.	DPM	CM	GP	PPC
3.	DPM	AIM		
4.	other	AIM		
5.	other	OPC		
6.	other	AIM		
7.	DPM	AIM		
8.	DPM	CM	GP	PPC
9.	DPM	CM	PG	GMR
10.	DPM	AIM		
11.	other	AIM		
12.	other	OPC		
14.	DPM	AIM		
15.	DPM	CM	GP	GMR
16.	DPM	AIM CM	MP	

Table 3 System trace: project creation

5.2 Communication during project execution

The unit manager receives requirements and specifications from the design project manager. This information is forwarded directly to the design engineer, the electrical engineer and the systems engineer. The communication patterns between team members is depicted in Figure 4.

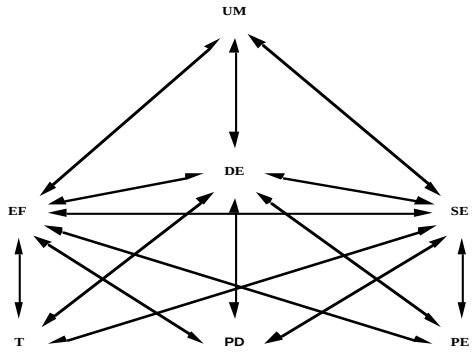

Figure 4 Communication patterns between design agents

The communication between team members includes information such as the position of the unit within the aircraft, the position of the door, supply lines, et cetera, but also customer requirements such as the size and number of towels that must fit in a cabinet. Requirements such as (fire) safety requirements, are not specified explicitly but are assumed to be known to the managers and engineers. In addition, the unit manager provides each engineer with relevant guidelines and planning information (e.g., deadlines and priorities). Guidelines, such as, "Use aluminum instead of stainless steel if at all possible", may evolve during the design process at unit management level. Such guidelines are forwarded immediately to the engineers - often causing modifications to existing (partial) designs.

The design engineer first analyses the information on the position of the unit. The initial contours of the unit and planes within the unit are identified. This initial sketch is given to the other engineers. This sketch roughly indicates where electrical, air-conditioning and water systems can be positioned. The electrical engineer and the systems engineer start working on a first draft of their systems, roughly following the priorities provided by the unit manager.

Expectations of the time involved in manufacturing guide the design strategy and thus scheduling of sub-tasks. The unit manager had initially given the toilet basin highest priority. The design engineer, however, expects the counter-top, sink and cabinet combination to be more complex. She informs the unit manager of her intention to work on the counter-top, sink and cabinet combination first, and the reasons for this decision. One of the reasons is the fact that the requirements differ considerably from previous designs, implying that extensive interaction with suppliers, product specialists and tooling specialists is required. The unit manager agrees with the argumentation and informs the other engineers of the change in priority.

The design engineer designs and positions the cabinets, the sink and the counter top. Different options are explored: properties of material, appearance, functionality, et cetera, are analysed in interaction with specialists. An example of the types of interaction involved is illustrated for the requirement that the overflow in the sink should not be immediately visible. This requirement mandates, in our example,
(1) interaction with the purchasing department to determine whether sinks exist for which the overflow is closer to the user than to the wall (so that it cannot be seen),
(2) interaction with the product specialist to determine whether and how a sink can be made to fulfil this requirement (if possible using standard components),
(3) interaction with the tooling specialist to determine whether specific tooling is required in the production process.

The design engineer discusses the different options with the systems engineer (position of the drain is of importance), and the electrical engineer (the position of the sensor to activate the water flow is of importance), and proposes a solution. If the unit manager agrees, the solution is accepted.

A similar pattern of communication is required for the counter top and the cabinets, in which case the styling expert is consulted for input on the precise shape of the combination. The process sketched above is described below in more detail, with a system trace as shown in Table 5.

Project execution: design scenario

During the design process for the counter top and the cabinets the component AST of the design engineer makes a partial initial design (1) which is sent by its AIM component (2) to the unit manager (UM), the electrical engineer (EE) and the systems engineer (SE).

The AIM components of the electrical and systems engineer (3) forward the initial design to their own AST components (4). The AIM components (5) of these agents then send the initial designs of their systems to the unit manager and the design engineer. Their AIM components (6) transfer these designs to the respective AST components (7). The electrical engineer sends an initial design of the electrical cable routing, the system design sends an initial design of all other systems.

The design engineer's AST component positions the electrical cable routing in her current design and discovers a problem: the cable routing directly crosses mounting points of the cabinet (7). Using her AIM component (8), the design engineer informs the electrical engineer and the unit manager of this problem. The information arrives in their AIM components (9) and is sent on to their AST components (10). The AST component of the electrical engineer solves the problem by re-routing the cable.The solution is sent by the AIM component (11) to the design engineer and the unit manager; they receive the solution in their AIM component (12). With the solution, the AST component of DE can resume its work (13).

To finalise the design of the counter top, sink and cabinet combination the AST component of the design engineer needs more detailed information on switches, light points, sensors, et cetera, from the electrical engineer, thus a request is sent by the design engineer's AIM component (14).

The design engineer also needs more detailed information on pipes and drains (size, mounting specifications, screws, et cetera) from the systems engineer, again a request is sent by the AIM component (14). In both cases the unit manager is informed as well (14). The AIM components of the EE, SE and UM receive the request (15).

The AST component of EE has to reschedule some of its sub-processes to provide this information as soon as possible (16). This is important for the ordering of the necessary materials and tooling. After rescheduling, the information is sent by EE's AIM to the design engineer (18).

In the mean time, the AST component of SE (16) is able to provide the information immediately, SE's AIM component (17) sends the information to DE. The design engineer receives the information from EE and SE, and via AIM (19) and AST (20)

proceeds to design the toilet basin, requiring interaction with both the electrical engineer and the systems engineer.

time point	agent	agent component
1.	DE	AST
2.	DE	AIM
3.	EE	AIM
	SE	AIM
4.	EE	AST
	SE	AST
5.	EE	AIM
	SE	AIM
6.	DE	AIM
	UM	AIM
7.	DE	AST
	UM	AST
8.	DE	AIM
9.	UM	AIM
	EE	AIM
10.	EE	AST
	UM	AST

time point	agent	agent component
11.	EE	AIM
12.	DE	AIM
	UM	AIM
13.	DE	AST
14.	DE	AIM
15.	EE	AIM
	SE	AIM
	UM	AIM
16.	EE	AST
	SE	AST
17.	SE	AIM
18.	EE	AIM
19.	DE	AIM
20.	DE	AST

Table 5 System trace: project execution

6 Discussion

Collaborative, concurrent engineering projects are complex. The coordination of these projects in virtual environments, in particular the coordination of conflicting (partial) designs, interests, models, requirements (e.g., new requirements imposed during design), et cetera, requires extensive knowledge of the design process, of the available expertise and skills, of dependencies and, in particular, of the consequences of modification. Recently a number of tools and services have been designed to support specific aspects of the coordination process; for example, (Bahler, Dupont and Bowen, 1994; Cutkosky, Engelmore, Fikes, Gruber, Genesereth, Mark, Tenenbaum and Weber, 1993; Klein, 1995; Petrie, 1994). In this paper a multi-agent perspective to project coordination is presented.

Multi-agent literature focusses on modelling interaction between agents, most frequently based on informal models of interaction; see (Wooldridge and Jennings, 1995). In this paper one of the models of agent cooperation (Jennings, 1995) has been formally specified, specialised and instantiated for an example scenario of a cooperative design project in which interaction is instantiated when necessary. The generic specifications of the model can be used in other project coordination situations, instantiated for the specific domain of application. By formally specifying not only the knowledge involved, but also the types of interaction and coordination patterns required during these types of projects, more detailed insight is acquired in the required type of support (for example, types of verification and validation).

Acknowledgements

Discussions on the cooperation model with Nick Jennings were very useful to the authors.

References

Bahler, D., Dupont, C., Bowen, J. (1994). Anaxiomatic approach that supports negotiated resolution of design conflicts in concurrent engineering. In: Gero, J.S., Sudweeks, F. (eds.), *Artificial Intelligence in Design*, Kluwer Academic Publishers, Dordrecht, pp. 363-379.

Brazier, F.M.T., B. Dunin-Keplicz, N.R. Jennings, J. Treur (1995). Formal Specification of Multi-Agent Systems: a Real World Case. In: V. Lesser (ed.), *Proc. of the First International Conference on Multi-Agent Systems, ICMAS-95*, MIT Press, pp. 25-32.

Brazier, F.M.T., B. Dunin-Keplicz, N.R. Jennings, J. Treur (1996). Modelling Distributed Industrial Processes in a Multi-Agent Framework. In: G. O'Hare, S. Kirns (eds.), *Towards the Intelligent Organisation: a Coordination Perspective*, Springer Verlag.

Brazier, F.M.T., C.M. Jonker & J. Treur (1996). Formal specification of a model for cooperation based on joint intentions. Technical Report, Vrije Universiteit Amsterdam, Department of Mathematics and Computer Science

Brazier, F.M.T., P.H.G. van Langen, Zs. Ruttkay & J. Treur (1994). On Formal Specification of Design Tasks. In: Gero, J.S. (ed.), *Artificial Intelligence in Design '94, Proceedings AID '94*. Kluwer Academic Publishers, Dordrecht, pp. 535-552.

Brazier, F.M.T. and J. Treur (1994). User centered knowledge-based system design: a formal modelling approach. In: L. Steels, G. Schreiber and W. Van de Velde (eds.), *A future for knowledge acquisition, Proceedings of the 8th European Knowledge Acquisition Workshop*, EKAW '94. Springer Verlag, Lecture Notes in AI 867, pp. 283-300.

Brazier, F.M.T., J. Treur, N.J.E. Wijngaards and M. Willems (1995). Formal specification of hierarchically (de)composed tasks. In: B.R. Gaines and M.A. Musen (eds). *Proceedings of the 9th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, KAW '95*, 1995, Volume 2, pp. 25/1-25/20. Calgary: SRDG Publications, Department of Computer Science, University of Calgary.

Brazier, F.M.T., J. Treur, N.J.E. Wijngaards and M. Willems (1996). Temporal semantics of complex reasoning tasks. In: R. Albrecht, H. Herre (eds). *Proc. of the Int. Workshop on New Trends in Theoretical Computer Science*, Oldenburg Verlag, Munich-Vienna.

Cutkosky, M., Engelmore, R., Fikes, R., Gruber, T., Genesereth, M., Mark, W., Tenenbaum, J., Weber, J. (1993). PACT: An experiment in integrating concurrent engineering systems. *IEEE Computer 26, Special Issue on Computer Support for Concurrent Engineering*, pp. 28-37.

Dunin-Keplicz, B. and J. Treur (1995). Compositional formal specification of multi-agent systems. In: (Wooldridge and Jennings, 1995), pp. 102-117

Dunskus, B.V., Grecu, D.L., Brown, D.C. and Berker, I. (1995). Using single function agents to investigate conflict. *AIEDAM 9*. Cambridge University Press, pp. 299-312

Jennings, N.R. (1995). Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems using Joint Intentions, *Artificial Intelligence Journal* 74 (2)

Klein, M (1995). Conflict management as part of an integrated exception handling approach. *AIEDAM 9*, Cambridge University Press, pp. 259-267.

Langevelde, I.A. van, A.W. Philipsen and J. Treur (1992). Formal specification of compositional architectures, in B. Neumann (ed.), *Proceedings of the 10th European Conference on Artificial Intelligence, ECAI'92* , John Wiley & Sons, Chichester, pp. 272-276.

Petrie, C. (1994). Design space navigation as a collaborative aid. In: Gero, J.S. (ed.), *Artificial Intelligence in Design '94, Proceedings AID '94.* Kluwer Academic Publishers, Dordrecht, pp. 611-623.

Wooldridge, M., Jennings, N.R. (eds.) (1995), *Intelligent Agents*, Proc. of the ECAI'94 Workshop on Agent Theories, Architectures and Languages, Lecture Notes in AI, vol. 890, Springer Verlag.